Demonstrating the Distinctions Between Persuasion and Deliberation Dialogues

Yanko Kirchev, Katie Atkinson, and Trevor Bench-Capon

Department of Computer Science,
University of Liverpool, UK
katie@liverpool.ac.uk

Abstract. A successful dialogue requires that the participants have a shared understanding of what they are trying to achieve, individually and collectively. This coordination can be achieved if both recognise the type of dialogue in which they are engaged. We focus on two particular dialogue types, action persuasion and deliberation dialogues, which are often conflated because they share similar speech acts. Previously, a clear distinction was made between the two in terms of the different pre- and post-conditions used for the speech acts within these dialogues. This prior work gave formal specifications of the dialogue moves within the dialogues but offered no evaluation through implementation. In this paper, we present an implementation to demonstrate that the two dialogue types described in this way can be realised in software to support focussed communication between autonomous agents. We provide the design and implementation details of our new tool along with an evaluation of the software. The tool we have produced captures the distinctive features of each of the two dialogue types, to make plain their differences and to validate the speech acts for use in practical scenarios.

Keywords: Argumentation, Dialogue, Deliberation, Persuasion

1 Introduction

Dialogues with computers are becoming an increasingly popular way of interacting with members of the public (e.g. [5], [7] and [11]) for giving advice, soliciting opinions, and e-participation generally. When modelling dialogues, it is important to be aware of the type of dialogue that one is dealing with. Dialogues are essentially cooperative acts between the participants, and it is therefore important that they are both playing by the same rules. These rules derive from the particular type of dialogue they are engaged in. Without a mutual understanding of the type of dialogue, participants will be at cross purposes, leading to misunderstandings and breakdown of the dialogue.

The notion of dialogue types can be found in [10], where five types were introduced. We will be concerned with two of these: persuasion and deliberation[1].

[1] Further in this paper, we will focus on persuasion and deliberation about *actions*. Although [10] might seem to suggest that persuasion concerns only propositions and

These two dialogues types are especially important in e-participation [2]. Persuasion is required to explain and defend policies, and deliberation is required in order to undertake public consultations. In [10], Walton and Krabbe characterise their dialogue types according to three aspects: the initial situation, the collective goal, and the individual goals of the participants.

- *Initial Situation*: For persuasion, the initial situation is a disagreement: the agents do not agree as to the best option. In deliberation, it is one of uncertainty: while the individuals may (or may not) have their own opinions about the best option, they do not know which option is collectively acceptable.
- *Collective Goal*: For both dialogue types, the collective goal is to come to an agreement as to what should be done, but there is a difference in what can be agreed. In persuasion, this is limited to the option proposed by the persuader being accepted by the persuadee; in deliberation, agreement can be to any option that is acceptable to the group as a whole. This is done by producing a rule expressing their collective preference, which can then be used to identify the most acceptable option for the group.
- *Individual Goals*: In persuasion, the individual goals are different: the persuader wishes to convince the other, whereas the persuadee, if cooperative, wishes to explore the possibility that its currently preferred option is not in fact best for it in the light of information known to the persuader. If uncooperative, the persuadee may wish to defend its own option or even convert the persuader. In a deliberation, the individual goals are the same for all participants: all participants wish to determine which option is the best for them collectively.

This characterisation does highlight some important differences between the dialogues. To an observer who has no access to the inner states of the participants, however, it may be difficult to tell which sort of dialogue is taking place. This is because the speech acts used are the same in both dialogue types. Although this is so, the force of the speech acts differs: the dialogue types determine how the acts are to be interpreted. The pragmatic effects of the various utterances differ: the *conversational implicatures* [6] of the utterances differ according to the dialogue type.

This aspect was explored in [1], where the various speech acts were described in terms of pre- and post-conditions. Each speech act has some of these common to both dialogue types, representing the semantic aspects of the act, but also additional different conditions for deliberation and persuasion which represent the pragmatic aspects of the act in the specific context. These pre-conditions show what should hold for the utterance to be legally made in the given dialogue type and what should be understood from the utterance in the different dialogues.

Although [1] gave full definitions of the pre- and post-conditions for the speech acts of persuasion and deliberation dialogues, there was no implementation, and hence no practical evaluation. Therefore, our aims were:

---

not actions, persuading people to do something is such an everyday occurrence that we may regard persuasion about action as a *bona fide* dialogue type.

– To implement and so evaluate the sets of speech acts proposed in [1] to demonstrate their adequacy or, if necessary, to refine them to provide an adequate set of speech acts;
– To provide a tool which will explicitly show the differences between the dialogue classes, allowing users to compare them and explore the different effects of the various speech acts.

The rest of the paper is structured as follows. In section 2, we discuss the various speech acts. The evaluation in fact showed that two speech acts additional to those given in [1] are required. Section 3 describes the design of the tool, and section 4 the realisation of this design. Section 5 describes how the tool can be used to input a situation and generate persuasion and deliberation dialogues, illustrated with the example from [1]. This example involves a scenario, expressed as a logic program, in which three agents are choosing a restaurant to dine out in and each agent has its own individual preferences. Our example dialogues show the differences in the commitment stores that result from the speech acts being deployed within the two different dialogue contexts. Section 6 shows the methods used to evaluate the tool, and section 7 offers some discussion and concluding remarks.

2 Speech Acts

The protocols for the two dialogue types are distinguished by specifying different pre- and post-conditions for the speech acts depending upon which of the two dialogues they are used within. Each agent is defined to have a 'commitment store' [10]: a set of statements to which they become publicly committed during the course of a dialogue. The pre-conditions determine the pre-requisites that need to be satisfied in terms of available knowledge and prior commitments of dialogue participants in order for the speech acts to be used legally. The post-conditions determine the updates on the agents' commitment stores that occur immediately after the enactment of the move. Our implementation initially follows [1] by using the speech acts set out in Prakken's dialogue system [9]. These speech acts are:

– *Claim*: used to assert a fact;
– *Why*: used to ask for a justification of a claim;
– *Since*: used to provide a justification for a claim;
– *Concede*: used to accept a claim;
– *Retract*: used to withdraw a claim;
– *Question*: used to seek a piece of information.

Prakken also sets out a protocol in the form of a set of rules by which a dialogue proceeds. The protocol specifies the speech acts permitted at any given point during the dialogue, the effects of utterances on the participants' commitments, the outcome of the dialogue, the turn-taking function, and the termination criteria. Our tool demonstrates the changes in the commitment stores of the agents as the dialogues proceed towards termination. The above set of speech acts is set out formally in [1].

2.1 Refinement of Speech Acts

Designing and implementing software to execute the persuasion and deliberation dialogues required the speech acts to be operationalised. Through designing the software, it became apparent that refinements were needed to two of the original speech acts from [1]. The *claim* speech act, used by opponents in the persuasion dialogue and all agents in the deliberation dialogue, turned out to be too restrictive; one of its pre-conditions allows for assertions to be made only about options that the agents find acceptable, thus rendering the agents incapable of making objections towards options that they do not find acceptable or of making claims about options they do not find acceptable.

A similar observation can be made about the *since* move in deliberation: some of its pre- and post-conditions are implied by, or clash with, other conditions. Moreover, deliberation changes the speech act's purpose to move previously proposed criteria into the body of the rule expressing the agreed preference, making agents in deliberation incapable of simply justifying previously made claims without changing the preference rule, which is the role of the *since* move in the other types of dialogues.

Given the above observations revealed through the implementation exercise, both the *claim* and *since* speech acts require modifications in order for the aforementioned difficulties to be overcome. The *claim* speech act has thus been modified by replacing its problematic pre-condition with a *preferable* function, which consequently allows opponents in persuasion and all agents in deliberation to make assertions about their initial preferences, even in the cases where they do not find them completely acceptable. Furthermore, a new speech act is introduced, *counterclaim*, which allows all types of agents to make objections against options that they do not find acceptable. Finally, the *since* speech act for deliberation has been split into two distinct acts: the general *since* move remains unchanged and is used to justify a previously made claim, but a new *concede-since* speech act is introduced that is only used to move previously proposed criteria into the body of the agreed preference rule.

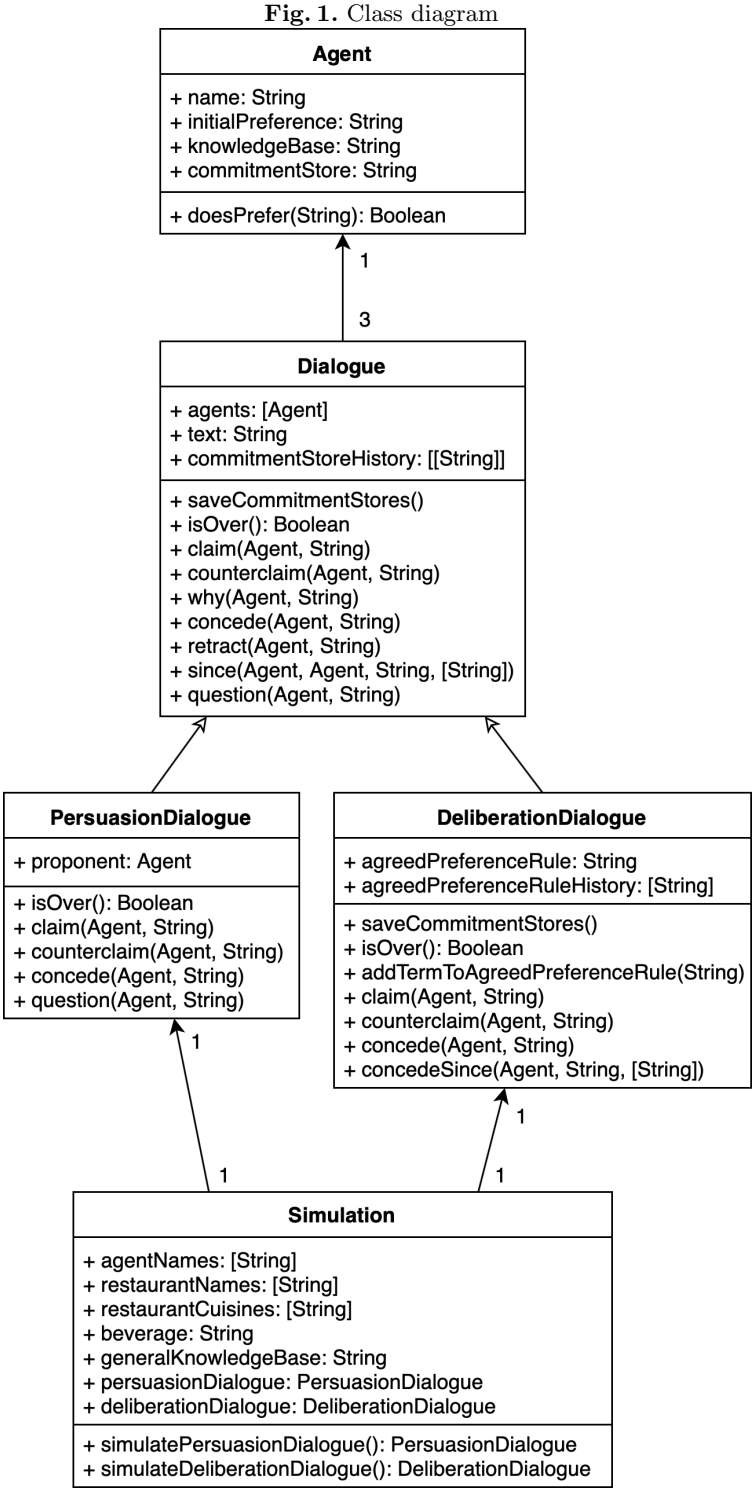
The refinements made to the speech acts, as described in this section, were found to be required to enable the sample dialogues given in [1] to be automated in an implementation. In the next section, we provide the design for the programs to realise the dialogues in software.

3 Design

The tool allows for the simultaneous generation of customisable persuasion and deliberation dialogues, which are set in the same restaurant selection scenario that was introduced in [1], where three agents try to decide on a place in which to dine. Consequently, the tool is called *Diners' Discourse*, and this section describes the object-oriented components used in the protocols.

The class diagram in Figure 1 illustrates the entities that the tool encapsulates and also provides insight on the relationships between the various objects that are created during a single lifecycle.

Fig. 1. Class diagram



3.1 Agent Class

The *Agent* class represents a participant in a dialogue, and every agent has the following attributes: a *name*; an *initialPreference*, which is a restaurant that the agent introduces to the dialogue as their preferred option regardless of whether they find it acceptable; a *knowledgeBase*, which is a static collection of Prolog facts and rules that specify the available knowledge of the agent about the world; and a *commitmentStore*, which is an initially empty set of statements that is populated with the statements to which the agent becomes committed in the course of a dialogue.

Alongside that, each *Agent* instance also includes a *doesPrefer* method, which corresponds to the *preferable* function discussed in the previous section, and which takes a restaurant name as an argument and returns *true* only if it matches the agent's initial preference or if the agent finds it acceptable.

3.2 Dialogue Class

The *Dialogue* class is abstract and represents a generic dialogue. Every dialogue has three participants stored in the *agents* attribute, and its conversational script is continuously expanded in the *text* attribute as the participants perform various moves.

The class also implements methods that correspond to the general specifications of the speech acts that are common to both persuasion and deliberation. In addition to that, it records the evolution of the commitment stores of the participating agents throughout the course of the dialogue by utilising the *saveCommitmentStores* method and the *commitmentStoreHistory* attribute. Moreover, the abstract *isOver* method returns a Boolean value that determines whether the dialogue is in an end state.

3.3 PersuasionDialogue Class

The *PersuasionDialogue* class extends the abstract *Dialogue* class and represents a persuasion dialogue. As such, it features an additional *proponent* attribute, which points to one of the objects in *agents* and which indicates the designated persuader of the dialogue.

The *isOver* method of the parent class receives a dialogue instance and returns *true* only if every agent is committed to the statement that the initial preference of the proponent is acceptable. Moreover, the *claim*, *counterclaim*, *concede*, and *question* methods are overridden to feature the additional pre- and post-conditions required by persuasion.

3.4 DeliberationDialogue Class

The *DeliberationDialogue* class extends the abstract *Dialogue* class and represents a deliberation dialogue. As such, it features an additional *agreedPreferenceRule* attribute, which represents the collective preference rule of what constitutes an acceptable restaurant that the agents construct during the course of

the dialogue, and its evolution is recorded in the *agreedPreferenceRuleHistory* attribute.

The class also overrides the *claim*, *counterclaim*, and *concede* methods to feature the additional pre- and post-conditions required by deliberation and also implements a *concedeSince* method, which corresponds to the deliberation-exclusive *concede-since* speech act. Moreover, a method *addTermToAgreedPreferenceRule* is utilised by the *concede* and *concedeSince* methods to add the locution used with them to the body of the agreed preference rule.

DeliberationDialogue also implements the *isOver* method of its parent class, and, in this case, it returns *true* only if all agents agree on the same option being acceptable and if the acceptability of the agreed option can be demonstrated from the collective commitment store of the participating agents in conjunction with the agreed preference rule, minus any private preference commitments.

3.5 Simulation Class

Simulation is a concept class that represents the lifecycle of one persuasion and one deliberation dialogue from their initialisation, through their execution, until they reach their final states. It contains a *generalKnowledgeBase* attribute, which is a refined template of the collective facts and rules of the participating agents from the model dialogues in [1]. The placeholder values in it are populated with the data from the *restaurantNames*, *restaurantCuisine*, and *beverage* attributes, and the populated general knowledge base along with the *agentNames* attribute is then used by the *createPersuasionDialogue* and *createDeliberationDialogue* methods to create three *Agent* objects each and to distribute the facts and rules amongst them.

Thereafter, *PersuasionDialogue* and *DeliberationDialogue* instances are created and predefined sequences of speech acts are performed on them. They are assigned to the appropriate *persuasionDialogue* and *deliberationDialogue* attributes after the *isOver* methods check that they have reached an end state.

In the next section, we cover the implementation details of the protocols and also present a web application that is provided as a user interface for the tool.

4 Realisation

The protocols are written in JavaScript and run on the Node.js environment with the help of the Tau Prolog module, which provides native knowledge representation without requiring Prolog itself. A graphical web interface allows for a user-friendly interaction with the tool and is constructed using the React.js framework together with Carbon Design's React components and supporting technologies such as HTML5 and CSS3.

4.1 Project Architecture

The project's software architecture is shown in Figure 2 and can also be further examined on its GitHub repository[2]. As can be seen, the front-end is written as modular components, each of which has a distinct purpose within the system and is contained within the *src* folder, whereas the *protocols* subfolder contains the back-end files, each of which corresponds to a class in Figure 1. Moreover, the protocols are supported by unit tests under *test*, which verify the soundness of the implementation, and additional helper files under *utils*.

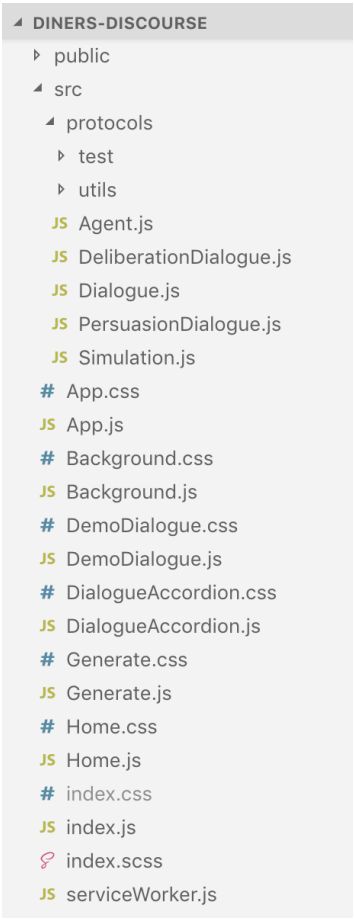


Fig. 2. Project architecture

[2] See: github.com/yankirchev/diners-discourse

4.2 Interface

The web interface of *Diners' Discourse* features a single, dynamic page, the view of which changes based on the selected tab of the content switcher in its top centre. The three available views are *Home*, which contains a welcome message and a demo dialogue; *Background*, which contains the purpose of, and the technologies used by, the application along with a glossary; and *Generate*, where the persuasion and deliberation dialogues are customised, generated, and presented to the users.

While this subsection focuses on the *Generate* tab, as it contains the main functionality of the application, the tool is available to be further examined in the public domain at the URL given in footnote 2.

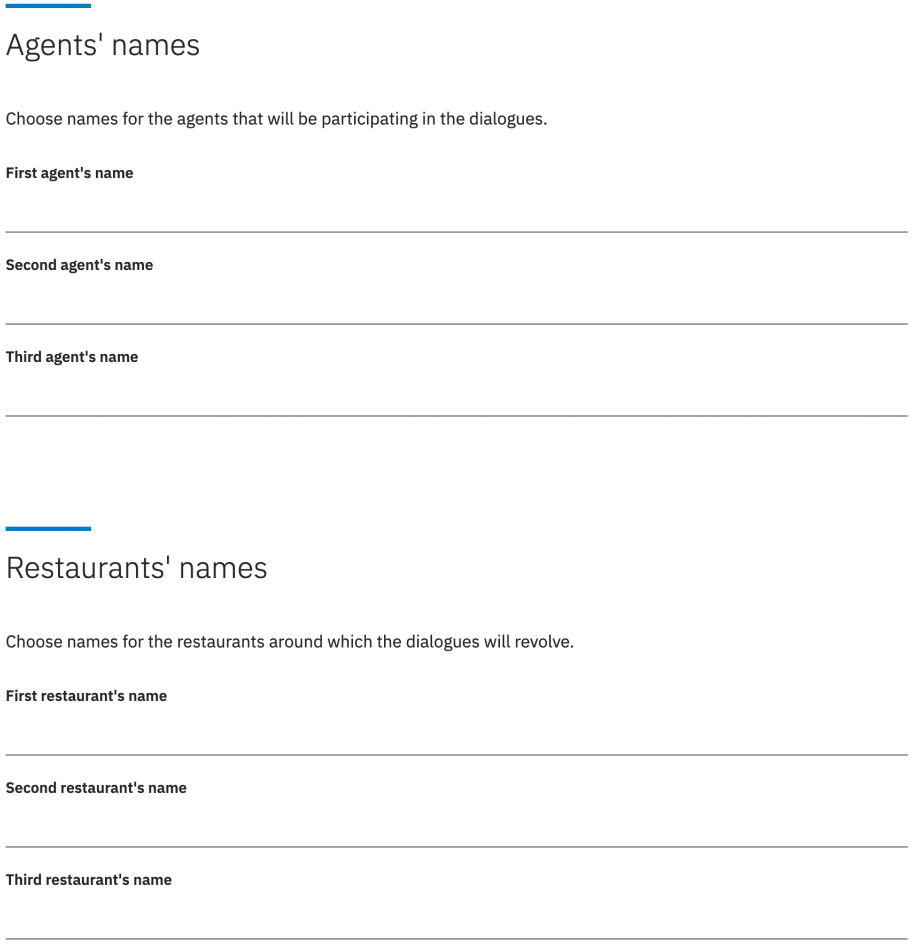


Fig. 3. First part of the input form

The *Generate* tab presents the users with an input form, as can be seen in Figures 3 and 4, which collects the required data for the customisation of the dialogues. The users are allowed to choose the names of the participating agents, the names and the cuisines of the restaurants, and the beverage that is preferred by the agents. The choices in regard to the names are entirely up to the users. However, for the cuisines of the restaurants and the beverage preferred by the agents, the choices are restricted. This restriction is so that the categories make sense in terms of the agent preferences, which are taken from [1].

The form also features a *Generate* button at the very bottom, which, when clicked, prompts *Diners' Discourse* to create a *Simulation* instance by populating the *agentNames*, *restaurantNames*, *restaurantCuisines*, and *beverage* attributes using the user input.

The next section showcases a persuasion dialogue and a deliberation dialogue generated by inputting *Jane*, *Harry*, and *George* as the agents' names; *La Zingara*, *Thai Palace*, and *Nosh* as the restaurants' names; *Italian*, *Thai*, and *American* as the restaurants' cuisines; and *wine* as the beverage preferred by the agents, as in the main example of [1].

5 Description of the Tool

The users are presented with the custom persuasion and deliberation dialogues side by side, as can be seen in Figure 5, which facilitates the comparison between the two types and therefore assists the users in identifying the distinctive features of persuasion and deliberation.

The dialogues are revealed speech act by speech act with the use of the *Next* buttons, which allow the users to gradually step through the dialogues without getting overwhelmed with too much information at once. Nonetheless, the full extent of the dialogues can be shown at any point by clicking the *Reveal all* buttons. Naturally, the *Back* buttons hide the last shown speech acts and the *Reset* button erases the current *Simulation* instance and starts over the customisation process again.

By clicking on any of the revealed speech acts, the users can view the state of the participating agents' commitment stores at that point in the dialogue. Moreover, they can have multiple speech acts of the same dialogue open, which assists them in understanding how the commitments evolve throughout the dialogues and how the use of specific speech acts affects commitments. Furthermore, by having speech acts of the two types expanded side by side, the users can gain insight into how moves such as *claim* and *question* have different effects on the commitment stores for persuasion and deliberation.

When a speech act of the deliberation dialogue is expanded, the body of the agreed preference rule is also shown alongside the state of the agents' commitments. This lets the users observe how the rule is constructed throughout the dialogue and how speech acts such as *concede* and *concede-since* determine the joint rule's final structure.

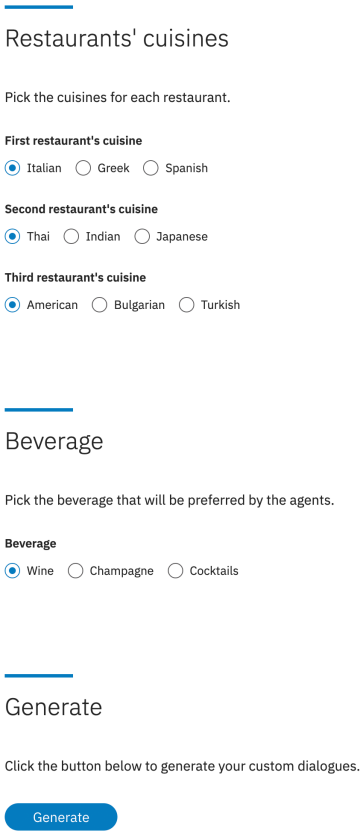


Fig. 4. Second part of the input form

6 Testing and Evaluation

The dialogues in Figure 5 closely adhere to those in [1] from which the protocols' formal definitions have been derived, indicating that the tool implements the dialogues as conceived in [1].

In order to demonstrate the close resemblance, we compare the structure of the two dialogues and the use of particular speech acts instead of their scripts, as the protocols do not use a natural language engine to produce the text of the dialogues. Ultimately, it is not essential that the dialogues match word for word, but rather that the same arguments by the same agents appear in both dialogues in the same context and that they reach the same conclusions.

While the resulting implemented dialogues feature some additional speech acts, these additions complement the general conduct of the dialogues and support the agents in arriving at their goals, which in turn make the generated persuasion and deliberation dialogues more complete versions of the model ones.

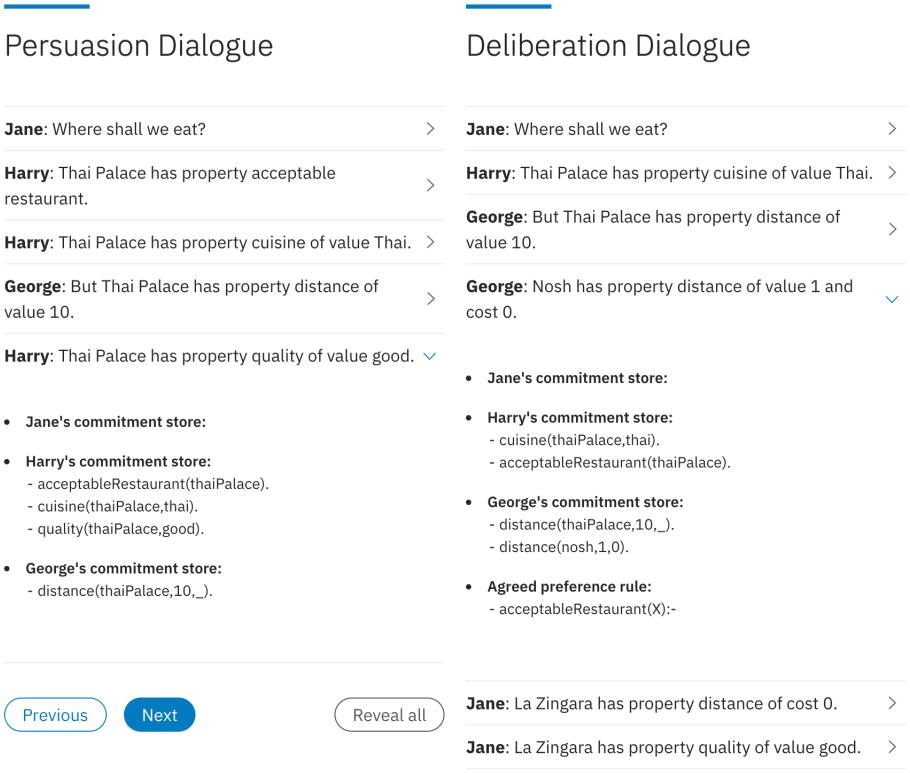


Fig. 5. Dialogues comparison

Aside from sounding less natural because of the computer-generated phrases, the produced dialogues are largely analogous to the model ones: in both sets of dialogues, the same agents have the same goals and execute the same moves towards achieving them. Moreover, the outcomes of the generated dialogues are the same as those of the model ones.

Ultimately, this corroborates the implemented protocols' integrity and effective application in a practical setting, meeting our aims for the work.

6.1 Unit Testing

In order to verify the correctness of the protocols' implementation, it is necessary to demonstrate that it exhibits the behaviour that the specifications of the speech acts describe for persuasion and deliberation. More precisely, we need to demonstrate that the pre-conditions of each speech act are adhered to when used in any situation and that the post-conditions occur thereafter as well.

Consequently, every pre-condition is implemented so that it throws an error with an identifying description when it is not satisfied, and a positive and a negative test is written for each. The positive test ensures that the pre-condition

does not throw an error when the speech act is used legally, whereas the negative test ensures that the pre-condition throws an error when the speech act is used illegally. The tests fail only if the speech acts exhibit behaviour that deviates from the expected results.

The post-conditions only have a positive test each: although they are called conditions, their function is to update the commitment stores of the agents. Therefore, their tests ensure that the commitments are amended as required given that the pre-conditions are satisfied. The tests fail only if the state of the commitment stores deviates from the expected results.

Thereafter, custom simulations of persuasion and deliberation dialogues are carried out separately for each pre- and post-condition, which are similar in function to the *Simulation* class described in subsection 3.5. A total of 125 unit tests were written using the Jest framework for JavaScript and all were passed successfully. The unit tests take just over a second to execute, with each of them averaging ∼11.13 milliseconds to run. This indicates the efficiency of the implemented protocols, which is an important factor in multi-agent communications where parties are expected to react quickly to an evolving situation.

The exhaustive number of unit tests along with their affirmative results corroborate the notion that any dialogues produced by the protocols are valid with respect to our expectations about persuasion and deliberation. Since each pre- and post-condition of every speech act is tested individually, the tests also ensure that the results of the conditions are independent of each other. Furthermore, the positive tests demonstrate that the legal use of the speech acts has the intended effect on the dialogue, while the negative tests show that no deviation from what is required of the speech acts is allowed.

7 Discussion and Concluding Remarks

We have described a tool that has been implemented to realise automated communication between several agents within action persuasion and deliberation dialogues. Our tool explicitly shows to users the differences between the dialogue types and enables exploration of the different effects of the various speech acts available. Whilst the software relied on the formal specification given in [1], the implementation exercise revealed refinements required to the underlying speech act specifications to enable accurate, realistic dialogues to be produced. The evaluation has demonstrated that the implementation exercise has been successful and we are encouraged that the distinctive features manifest in the two types of dialogue can be captured in tools for automated communication.

In implementing the two dialogue types, we have extended the speech acts based on [9] that were used in [1] to include two additional acts tailored for these dialogue types. We have shown that these speech acts enable dialogues of both types to be conducted. They can therefore form the basis of further implementations designed to address particular e-participation tasks. These tasks include the presentation and justification of policies, which require the user to be persuaded, and deliberations such as consultation about the suitability of a

range of options, intended to gauge which options have public support, and to enable progress towards a consensus about the preferred option. Thus, although the particular tool described here is intended primarily to teach the differences between to two types of dialogue, the implementation can form the basis for the realisation of a range of e-participation applications.

Future work will focus on two aspects. On a practical level, we will adapt the tool so that the speech acts can be used to implement some specific tasks, such as the e-participation tasks mentioned above. Of course, any fielded application of such dialogue agents would need careful consideration of the ethical issues, particularly as "fake news" and campaigns of disinformation become more prevalent. Consideration of ethical issues associated with conversational agents are discussed in [8]. At a more theoretical level, we will investigate how the speech acts of the dialogue types discussed here relate to other dialogue types, such as inquiry [4] and examination [3].

References

1. Atkinson, K., Bench-Capon, T., Walton, D.: Distinctive features of persuasion and deliberation dialogues. Argument & Computation **4**(2), 105–127 (2013)
2. Bench-Capon, T., Atkinson, K., Wyner, A.: Using argumentation to structure e-participation in policy making. In: Transactions on Large-Scale Data-and Knowledge-Centered Systems XVIII, pp. 1–29. Springer (2015)
3. Bench-Capon, T., Doutre, S., Dunne, P.E.: Asking the right question: forcing commitment in examination dialogues. In: Besnard, P., Doutre, S., Hunter, A. (eds.) Proceedings of COMMA 2008. vol. 172, pp. 49–60. IOS Press (2008)
4. Black, E., Hunter, A.: An inquiry dialogue system. Autonomous Agents and Multi-Agent Systems **19**(2), 173–209 (2009). https://doi.org/10.1007/s10458-008-9074-5
5. Chalaguine, L.A., Hamilton, F.L., Hunter, A., Potts, H.W.: Argument harvesting using chatbots. In: Proceedings of COMMA 2018. pp. 149–160. IOS Press (2018)
6. Grice, H.P.: Logic and conversation. In: Cole, P., Morgan, J.L. (eds.) Syntax and Semantics, Vol. 3, pp. 41–58. Academic Press: New York (1975)
7. Morgan, J., Paiement, A., Seisenberger, M., Williams, J., Wyner, A.: A chatbot framework for the children's legal centre. In: Proceedings of JURIX 2018. pp. 205–209 (2018)
8. Olszewska, J.I., Houghtaling, M., Gonçalves, P., Haidegger, T., Fabiano, N., Carbonera, J.L., Fiorini, S.R., Prestes, E.: Robotic ontological standard development life cycle. In: IEEE International Conference on Robotics and Automation 2018: workshop on Elderly Care Robotics: Technology and Ethics (2018)
9. Prakken, H.: Formal systems for persuasion dialogue. The Knowledge Engineering Review **21**(2), 163–188 (2006)
10. Walton, D., Krabbe, E.: Commitment in dialogue: Basic concepts of interpersonal reasoning. SUNY Press (1995)
11. Wardeh, M., Wyner, A., Atkinson, K., Bench-Capon, T.: Argumentation based tools for policy-making. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law. pp. 249–250. ACM (2013)